

CLAIMS

I/We claim:

- [c1] 1. A method in a computer system for preparing a task to be swapped out from processor utilization, the computer system having multiple processors and an operating system, each processor having multiple streams for simultaneously executing threads of the task, the task having one or more teams of threads, each team representing threads executing on a single processor, the method comprising:
- raising an exception for each stream of each processor currently executing a thread of the task; and
 - in response to the raising of the exception, for each stream executing a thread,
 - saving a state of the stream;
 - determining whether the stream is a team master stream;
 - when the stream is not the team master stream, quitting the stream;
 - when the stream is the team master stream,
 - waiting for all other streams executing threads in the same team to quit;
 - determining whether the stream is a task master stream,
 - when the stream is not the task master stream, notifying the operating system that the team for this processor is ready to be swapped out;
 - when the stream is the task master stream,
 - waiting for all other teams to notify the operating system that the team is ready to be swapped out; and
 - notifying the operating system that the task is ready to be swapped out.

- [c2] 2. The method of claim 1 wherein the team master stream is a stream that increments a team master variable first.
- [c3] 3. The method of claim 1 wherein the task master stream is a stream that first notifies the operating system that its team is ready to be swapped out.
- [c4] 4. The method of claim 1 wherein the task master stream is a stream that last notifies the operating system that its team is ready to be swapped out.
- [c5] 5. The method of claim 1 wherein the state of the stream is stored in a list of stream states.
- [c6] 6. A system for preparing a task to be swapped out from processor utilization, the system having multiple processors and an operating system, each processor having multiple streams for executing threads of the task, the task having one or more teams of threads, each team representing threads executing on a single processor, each thread having a state, the system comprising:
- a component that raises an exception for each stream of each processor currently executing a thread of the task; and
 - an exception handler for each thread that, in response to the raising of an exception for the stream upon which the thread is executing,
 - saves the state of the thread;
 - determines whether it is a team master stream;
 - when the stream is not the team master stream, quits;
 - when the stream is the team master stream,
 - waits for all other streams executing threads in the same team to quit;
 - determines whether it is a task master stream; and
 - when it is not the task master stream, notifies the operating system that the team for this processor is ready to be swapped out; and

when the stream is the task master stream,

waits for all other teams to notify the operating system that
the team is ready to be swapped out; and

notifies the operating system that the task is ready to be
swapped out.

[c7] 7. The system of claim 6 wherein the team master stream is a stream
that increments a team master variable first.

[c8] 8. The system of claim 6 wherein the task master stream is a stream
that first notifies the operating system that its team is ready to be swapped out.

[c9] 9. The system of claim 6 wherein the task master stream is a stream
that last notifies the operating system that its team is ready to be swapped out.

[c10] 10. The system of claim 6 wherein the state of the stream is stored in a
list of stream states.

[c11] 11. A method in a computer system for preparing a task to be swapped
out from processor utilization, the computer system having multiple processors
and an operating system, each processor having multiple streams for executing
threads of the task, the task having one or more teams of threads, each team
representing threads executing on a single processor, the method comprising:

for each team, designating one stream that is executing a thread as a team
master stream;

designating one stream that is executing a thread as a task master stream
for the task;

for each team master stream, notifying the operating system that the team
is ready to be swapped out when each other thread of the team has
quit its stream; and

for the task master stream, notifying the operating system that the task is ready to be swapped when each of the other teams have notified the operating system that that team is ready to be swapped out.

[c12] 12. The method of claim 11 wherein the operating system swaps out the task upon receiving the notification that the task is ready to be swapped out.

[c13] 13. The method of claim 11 wherein each stream stores its own state before quitting the stream or notifying the operating system.

[c14] 14. The method of claim 11 wherein each stream that is not a team master stream quits its stream.

[c15] 15. The method of claim 11 wherein the notifying of the operating system by the task master stream includes indicating whether the task is blocked so that the operating system can defer swapping in the task until an event occurs to unblock the task.

[c16] 16. The method of claim 11 wherein each team master stream notifies the operating system of the number of streams that were executing threads so that the operating system can defer swapping in the task until enough streams are available to execute each of the threads that were executing when the task was swapped out.

[c17] 17. A system for preparing a task to be swapped out from processor utilization, the system having multiple processors and an operating system, each processor having multiple streams for executing threads of the task, the task having one or more teams of threads, each team representing threads executing on a single processor, the system comprising:

a component that, for each team,

designates one stream that is executing a thread as a team master stream; and

designates one stream that is executing a thread as a task master stream for the task;

a component that, for each team master stream, notifies the operating system that the team is ready to be swapped out when each other thread of the team has quit its stream; and

a component that, for the task master stream, notifies the operating system that the task is ready to be swapped out when each of the other teams have notified the operating system that that team is ready to be swapped out.

[c18] 18. The system of claim 17 wherein the operating system swaps out the task upon receiving the notification that the task is ready to be swapped out.

[c19] 19. The system of claim 17 wherein each stream stores its own state before quitting the stream or notifying the operating system.

[c20] 20. The system of claim 17 wherein each stream that is not a team master stream quits its stream.

[c21] 21. The system of claim 17 wherein the component that notifies the operating system that the task is ready to be swapped indicates whether the task is blocked so that the operating system can defer swapping in the task until an event occurs to unblock the task.

[c22] 22. The system of claim 17 wherein each team master stream notifies the operating system of the number of streams that were executing threads so that the operating system can defer swapping in the task until enough streams are available to execute each of the threads that were executing when the task was swapped out.

[c23] 23. A method in a computer system for preparing a task to be swapped out from processor utilization, the computer system having a processor and an operating system, the method comprising:

saving state information of each stream of the processor that is executing a thread;

under control of each stream that is not a master stream, quitting the stream; and

under control of the master stream,

waiting for each stream that is not a master stream to quit; and

providing a notification that the task is ready to be swapped out.

[c24] 24. The method of claim 23 wherein the operating system swaps out the task upon receiving the notification that the task is ready to be swapped out.

[c25] 25. The method of claim 23 wherein each stream saves its own state information.

[c26] 26. The method of claim 23 including upon being swapped in:
starting execution of a master stream; and
under control of the master stream,
creating a stream corresponding to each stream that quit when the task was swapped out; and
initializing each created stream based on state information saved before the stream quit.

[c27] 27. The method of claim 23 wherein the notification includes an indication of whether the task is blocked so that swapping in of the task can be deferred until an event occurs to unblock the task.

[c28] 28. The method of claim 23 wherein the task notifies the operating system of the number of streams that were executing threads so that swapping in

of the task can be deferred until enough streams are available to execute each of the threads that were executing when the task was swapped out.

[c29] 29. A system for preparing a task in a computer system to be swapped out, the computer system having a processor and an operating system, the processor having streams for executing threads of the task, each stream having a state, the system comprising:

means for saving the state of each stream that is executing a thread;

means for quitting each stream that is not a master stream; and

means for, when the stream is the master stream,

waiting for each stream that is not a master stream to quit; and

notifying the operating system that the task is ready to be swapped out.

[c30] 30. The system of claim 29 wherein the operating system swaps out the task upon receiving the notification that the task is ready to be swapped out.

[c31] 31. The system of claim 29 wherein each stream saves its own state.

[c32] 32. The system of claim 29 including upon being swapped in:

means for starting execution of a master stream; and

means for, when the stream is the master stream,

creating a stream corresponding to each stream that quit when the task was swapped out; and

initializing each created stream based on state information saved before the stream quit.

[c33] 33. The system of claim 29 wherein the means for notifying the operating system by the master stream includes means for indicating whether the task is blocked so that the operating system can defer swapping in the task until an event occurs to unblock the task.

- [c34] 34. The system of claim 29 including means for notifying the operating system of the number of streams that were executing threads so that the operating system can defer swapping in the task until enough streams are available to execute each of the threads that were executing when the task was swapped out.
- [c35] 35. A method in a computer system for restarting execution of a task that has been swapped out of processor utilization, the method comprising:
 starting execution of a master stream; and
 under control of the master stream,
 creating a stream corresponding to each stream that quit when the
 task was swapped out; and
 initializing each created stream based on state information saved
 when the stream quit.
- [c36] 36. The method of claim 35 wherein the method is performed by user runtime code.
- [c37] 37. The method of claim 35 wherein the starting is done in response to a signal.
- [c38] 38. The method of claim 37 wherein the signal is generated at the end of a time interval.
- [c39] 39. The method of claim 37 wherein the signal is generated when a routine is invoked.
- [c40] 40. The method of claim 35 wherein the restarting is deferred until a sufficient number of streams is available.

[c41] 41. A method in a computer system for restarting execution of a task that has been swapped out of processor utilization, the method comprising:
starting execution of a task master stream;
initializing the task master stream based on information saved when the task master stream quit;
under control of the task master stream, starting team master streams; and
under control of each team master stream,
creating a stream corresponding to each stream associated with the master stream that quit when the task was swapped out; and
initializing each created stream based on state information saved when the stream quit.

[c42] 42. The method of claim 41 wherein the streams associated with a team master stream were executing on the same processor when the streams quit.

[c43] 43. The method of claim 41 wherein each team master stream executes on a different processor.

[c44] 44. A method in a computer system for preparing a task to be swapped out from utilization of processors, the method comprising:
saving state information for each stream of the task; and
for each processor,
when no stream of the task executing on the processor is a task master stream,
providing notification that the streams of the processor are ready to be swapped out; and
when a stream of the task executing on the processor is a task master stream,
waiting for a stream of each other processor to provide notification; and
providing notification that the task is ready to be swapped.

- [c45] 45. The method of claim 44 wherein the task master stream is a stream that first notifies the operating system that it is ready to be swapped out.
- [c46] 46. The method of claim 44 wherein the task master stream is a stream that last notifies the operating system that it is ready to be swapped out.
- [c47] 47. The method of claim 44 wherein the state of the stream is stored in a list of stream states.